

日本国特許庁
JAPAN PATENT OFFICE

DN. 0111-00000000
K. OKADA 7-22-4
Filed 4-22-04
BSKB 703-205-8000

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 2003年 4月24日
Date of Application:

出願番号 特願2003-120600
Application Number:
[ST. 10/C]: [JP 2003-120600]

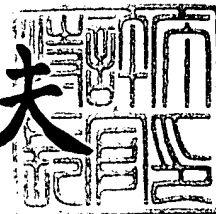
出願人 シャープ株式会社
Applicant(s):

CERTIFIED COPY OF
PRIORITY DOCUMENT

2003年12月16日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫



出証番号 出証特2003-3104413

【書類名】 特許願

【整理番号】 03J00997

【提出日】 平成15年 4月24日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 17/50

【発明者】

【住所又は居所】 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号 シャープ株式会社内

【氏名】 岡田 和久

【特許出願人】

【識別番号】 000005049

【氏名又は名称】 シャープ株式会社

【代理人】

【識別番号】 100078282

【弁理士】

【氏名又は名称】 山本 秀策

【選任した代理人】

【識別番号】 100062409

【弁理士】

【氏名又は名称】 安村 高明

【選任した代理人】

【識別番号】 100107489

【弁理士】

【氏名又は名称】 大塩 竹志

【手数料の表示】

【予納台帳番号】 001878

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0208587

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 動作合成システム、動作合成方法、制御プログラム、可読記録媒体、論理回路の製造方法および論理回路

【特許請求の範囲】

【請求項 1】 ループ処理および非ループ処理を含む動作記述から、各種処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフロログラフを生成するコントロールデータフロログラフ生成手段と、該コントロールデータフロログラフを用いてレジスタトランスファレベルのハードウェア構成を自動合成する手段とを有した動作合成システムにおいて、

該コントロールデータフロログラフ生成手段は、該ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の 1 回毎に該各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフロログラフのループ処理部内に、該ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれステージ毎の各節点にそれぞれ出力可能とするループ制御部が設けられた動作合成システム。

【請求項 2】 前記ループ制御部は、前記各制御信号をそれぞれ前記ステージ毎の各節点にそれぞれ出力する複数のポートと、該複数のポートのうち所定の一または複数のポートに接続されて終了判定を行う終了条件比較節点とを有する請求項 1 または 2 記載の動作合成システム。

【請求項 3】 前記ループ処理部は、所定の入力初期値が入力され、これを 1 サイクル毎にインクリメントした値を順次出力するループ用ポートと、該ループ用ポートからの出力値に対してループ処理の回数が規定回数に達したかどうかを判定する判定出力を前記ループ制御部に出力するループ回数判定節点と、前記ステージ毎の各節点とを有する相請求項 1 または 2 に記載の動作合成システム。

【請求項 4】 前記ループ制御部は、前記ループ回数判定節点からの判定出力を受けて、該ループ制御部の各ポートから前記ステージ毎の各節点にそれぞれ、非ループ処理を行わせる各制御信号をそれぞれ出力する請求項 3 に記載の動作合成システム。

【請求項 5】 前記ループ制御部の複数のポートは、前記パイプライン化ス

ページのステージ段数と同じビット数のシフトレジスタの機能を有している請求項 2 に記載の動作合成システム。

【請求項 6】 前記複数のポートは、前記ステージ毎の各節点にそれぞれ、前記シフトレジスタ機能によって所定の入力初期値が 1 サイクル毎に順次シフトされた各制御信号をそれぞれ供給する請求項 5 に記載の動作合成システム。

【請求項 7】 前記ステージ毎の各節点であって、前記ループ処理部の外部に対して作用する節点に対して、当該節点を前記ループ制御部からの制御信号により制御する請求項 1 に記載の動作合成システム。

【請求項 8】 前記コントロールデータフロログラフ生成手段は、パイプライン化されていないループ処理のコントロールデータフロログラフに基づいて、パイプライン化されたループ処理のコントロールデータフロログラフを生成する請求項 1 に記載の動作合成システム。

【請求項 9】 前記コントロールデータフロログラフ生成手段は、前記パイプライン化されていないループ処理のコントロールデータフロログラフに対して、ステージ境界部分とグラフの枝が交差する箇所に新たにポートを追加するポート追加手段と、並列処理を示すために各ステージを横方向に配置する横方向配置手段と、横方向に配置されたループ処理部の各ステージに対して、繰り返し間データ転送枝を接続処理する枝接続手段と、該ループ処理部内に前記ループ制御部を設けるループ制御部追加手段とを有する請求項 8 に記載の動作合成システム。

【請求項 10】 前記枝接続手段は、ループ処理部外から与えられる初期値およびループ処理部内で演算された値の何れか一方を選択してループ処理部内の変数に代入するセクタ節点を生成し、前記ループ制御部追加手段は、何れの値が代入されるかが前記ループ制御部の制御信号によって制御されるように、該ループ制御部とセクタ節点とを接続処理する請求項 9 に記載の動作合成システム。

【請求項 11】 ループ処理および非ループ処理を含む動作記述から、演算処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフロログラフを生成するコントロールデータフロログラフ生成工程と、該コントロールデータフロログラフを用いてレジスタトランスファレベルのハードウェア

構成を自動合成する工程とを有する動作合成方法において、

該コントロールデータフロウグラフ生成工程は、該ループ処理に含まれる各節点を各パイプライン化ステージ毎に分ける工程と、複数回のループ処理の 1 回毎に該各パイプライン化ステージの処理を並行して実行するループ処理部を生成する工程と、該ループ処理部内に、該ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を該各節点に出力可能とするループ制御部を生成する工程とを有する動作合成方法。

【請求項 1 2】 請求項 1 1 に記載の動作合成方法の各処理手順をコンピュータに実行させる制御プログラム。

【請求項 1 3】 請求項 1 2 に記載の制御プログラムが記録されたコンピュータ読み取り可能な可読記録媒体。

【請求項 1 4】 請求項 1 に記載の動作合成システムにより自動合成された回路構成に基づいて論理回路を製造する論理回路の製造方法。

【請求項 1 5】 複数の論理演算手段による並列演算を繰り返し行うループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれ複数の論理演算手段の少なくとも何れかに出力可能とするループ制御部が設けられた論理回路。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、例えば論理回路などの回路設計に用いられ、ループ処理を含む動作記述からパイプライン化されたコントロールデータフロウグラフを生成する動作合成システム、これを用いて R T L (Register Transfer Level ; レジスタトランスファレベル) の回路を自動合成する動作合成方法、その各処理手順をコンピュータに実行させる制御プログラム、これが記録されたコンピュータ読み取り可能な可読記録媒体、それらを用いた論理回路の製造方法および論理回路に関する。

【0 0 0 2】

【従来の技術】

従来、論理回路を含むシステム L S I などの大規模回路の設計において、動作記述から R T L の論理回路を生成する動作合成処理が行われている。

【0003】

この動作合成処理は、高位合成処理とも言われており、ハードウェアの構造に関する情報が含まれず、各処理のアルゴリズムのみが記述された動作記述から、R T L レベルのハードウェア（回路図）が自動的に合成される。

【0004】

例えば、非特許文献 1 には、C 言語をハードウェア設計用に拡張した言語を動作記述言語として、ハードウェア（回路図）を合成することが可能なコンピュータシステムである動作合成システムが開示されている。

【0005】

また、非特許文献 2 には、従来の動作合成技術の詳細が記載されており、コントロールデータフローグラフ（C D F G）から所望のハードウェア（回路図）を得る処理の詳細について記載されている。

【0006】

以下に、従来の動作合成技術を用いて、動作記述から所望の回路図を自動的に合成する動作合成装置による動作合成方法（高位合成方法）について簡単に説明する。

【0007】

この動作合成装置は、コンピュータシステムにて構成され、例えば、後述する図示しない C D F G 生成手段と、スケジューリング処理手段と、アロケーション処理手段と、データパス生成手段と、コントローラ生成手段とを有しており、図 10 の動作合成方法の各処理手順を順次行うことにより R T L（レジスタトランスファレベル）のハードウェア（回路図）を自動合成（自動設計）する。

【0008】

図 10 に示すように、まず、ステップ S 1 のコントロールデータフローグラフ（C D F G）生成では、C D F G 生成手段がアルゴリズム記述（動作記述）におけるデータの流れを解析し、C D F G と称されるモデルを作成する。C D F G は、演算を表す節点と、データの流れを表す枝とによって構成されている。各節点

は、入力枝と出力枝とによって接続されており、入力枝は演算に与えられるデータ、出力枝は演算結果のデータを表す。また、各節点はそれぞれ、演算の種類に関する情報も有している。

【0009】

例えば図11に示すC言語で記述された動作記述は、図12に示すCDFGによって表される。

【0010】

図12のCDFGは、二つの乗算を表す節点1および2と、一つの加算を表す節点3とを含み、節点1で入力データaおよびbが乗算された結果と、節点2で入力データbおよびcが乗算された結果とが、節点3で加算され、その結果が出力データXとして出力されることを表している。

【0011】

このCDFGは、コンピュータ（計算機）上では、例えば図13に示すようなデータ構造で表される。

【0012】

図13のデータ構造では、節点はNode構造体で表され、各節点固有の節点番号node__idを有している。また、in__edge配列には入力枝の枝番号が、out__edge配列には出力枝の枝番号が、複数記憶される。図13の例に示される節点は2入力1出力の演算であり、in__edgeは二つの要素、out__edgeは一つの要素を有している。また、op__typeには加算・減算・乗算などの各種演算の種類を表す番号が記憶されている。

【0013】

また、枝はEdge構造体で表され、各枝固有の枝番号edge__idを有している。from__nodeおよびto__nodeには、その枝が接続された節点の節点番号が格納されている。

【0014】

図13のようなデータ構造により、コンピュータ（計算機）上のメモリ内にCDFGの各節点間の接続が記憶されている。

【0015】

また、C D F G上で節点を接続する場合や、ある節点の入出力につながる別の節点を見つける場合には、以上のようにして計算機上のメモリ内に記憶されたデータの各要素に節点や枝の番号を登録し、またはそれを参照する。

【0016】

以下では、説明を分かり易くするために、図12のC D F Gを視覚的に表し、節点を枝で接続したり、節点に枝で接続された別の節点を参照しながら説明を行う。

【0017】

次に、図10のステップS2およびステップS3の各処理では、ステップS1で生成されたC D F Gに対して、スケジューリング処理手段がスケジューリング処理を行い、さらに、アロケーション処理手段がアロケーション処理を行う。

【0018】

スケジューリング処理では、C D F G中の各節点をいつ実行させるかを決定する処理である。スケジューリング処理は、C D F G中の各節点がいくつかのステップに分けられる。一つのステップに含まれる節点は、クロックの変化の1サイクル中に実行される。

【0019】

アロケーション処理はバインディング処理とも言われ、C D F G中の枝で表されるデータを格納するレジスタを決定する処理と、C D F G中の節点で表される演算を、どの演算器を用いて行うかを決定する処理とからなっている。動作合成方法（高位合成方法）によっては、このアロケーション処理がスケジューリング処理の前に行なわれる場合もある。

【0020】

次に、図10に示すように、ステップS4およびステップS5の各処理では、スケジューリング結果およびアロケーション結果を元に、データパス生成手段がデータパスを生成し、さらに、コントローラ生成手段がコントローラを生成することによりR T L（レジスタトランスファレベル）のハードウェア（回路図）が自動合成（自動設計）される。このように、C D F Gからハードウェア（回路図）を得る処理の詳細については上記非特許文献2に記載されており、また、その

一例が特許文献 1 に記載されている。

【0021】

ところで、実際の設計回路では、動作記述中にループ処理が含まれていることが多い。このループ処理は、多数回繰り返して実行されるため、回路全体の処理時間のうち、多くは、ループ処理によって占められる。

【0022】

したがって、回路の合成処理を高速化したい場合には、このループ処理を高速化することが有効である。例えば、画像処理や音声処理などのように、ある一定の時間内に処理を終える必要があり、高速性が要求される回路設計の場合には、このようなループ処理の高速化が必須である。

【0023】

このようなループ処理の高速化という問題を解決する方法の一つとして、ループ処理をパイプライン化する方法が挙げられる。

【0024】

従来のループ処理のパイプライン化方法としては、例えば非特許文献 3 に開示されているような方法が挙げられ、この方法は例えば特許文献 2 の図 3 (e) に図示されている。

【0025】

以下に、従来のループのパイプライン化方法について、図 14 に示すようなループ処理を含む動作記述を一例として簡単に説明する。図 14 では、 $i = 0$ から i を一つずつインクリメントしながら $i \leq 10$ の条件を満たす間、 $f(i)$ 、 $g(i)$ および $h(i)$ の演算を行うという各処理を表す動作記述である。

【0026】

図 15 は、図 14 に示す動作記述について、ループ処理をパイプライン化しない場合の C D F G を示す図である。

【0027】

図 15 において、矩形 10 はループ処理部を表し、ループ処理部 10 内の C D F G が繰り返し実行される。ループ処理部 10 の上端と下端に示す丸印 12 および 13 はループ処理部 10 のポートを表し、図 15 の変数 i に対応する。ポート

12に格納された変数*i*の値（データ）がインクリメント演算節点14でインクリメントされて一つ増加するように変化し、ポート13に至る。ポート12と13とは同じ変数を表しており、ポート13に至ったデータは次の繰り返しで再びポート12に戻されて使用されるようになっている。ループ処理部10の外の節点11は定数を入力する節点であり、まず変数*i*の初期値がポート12に与えられる。

【0028】

図14の動作記述では、変数*i*のみがループ処理部10内で繰り返し使用されるが、例えば2個以上の変数がループ処理部内で繰り返し使用される場合には、12および13のようなポートの対が、変数の個数分だけ設けられることになる。

【0029】

節点15では終了条件の比較が行われ、「真 ($i \leq 10$)」であれば「1」、「偽」であれば「0」が出力される。

【0030】

節点16はループ処理部10の終了を制御する特別な「EXIT」節点であり、終了条件比較節点15からの入力データが「0」の場合（変数*i*が10以下の場合）には、図示しないコントローラに対してループ10の処理を終了し、次の状態に移行するように指示が行われる。

【0031】

節点17、18および19ではそれぞれ、図14の動作記述中の関数*f*、*g*および*h*の各種演算処理が行われる。それぞれの節点17、18および19には、変数*i*の値が入力されている。節点17、18および19はそれぞれ、ステップ1、2および3にスケジューリングされている。

【0032】

上記構成により、その処理動作を説明すると、クロックの1サイクル目で、1回目のループ処理のステップ1に含まれる節点17で演算*f*、即ち*f*（1）が実行され、2サイクル目で、1回目のループ処理のステップ2に含まれる節点18で演算*g*、即ち*g*（1）が実行され、3サイクル目で、1回目のループ処理のス

テップ 3 に含まれる節点 19 で演算 h 、即ち $h(1)$ が実行される。

【0033】

次に、4、5 および 6 サイクル目で、2 回目のループ処理のステップ 1、2 および 3 に含まれる節点 17、18 および 19 で $f(2)$ 、 $g(2)$ および $h(2)$ がそれぞれ順次実行される。したがって、ループ処理を 10 回繰り返すためには、30 サイクルの処理が必要となる。

【0034】

なお、C D F G でループ処理を表す方法には、決まった形式がなく、各文献毎に様々な表記がされているが、動作としては同一である。

【0035】

次に、図 14 の動作記述のループ処理をパイプライン化する場合について説明する。

【0036】

ここでは、ループ処理のパイプライン化段数は 3 段とする。したがって、ループ処理を 1 回繰り返すときの処理を 3 ステージに分ける。なお、ステージとは、パイプライン動作をする場合の処理のまとまりのことである。

【0037】

図 14 の動作記述のループ処理は、パイプライン化されない場合には、1 回の繰り返しが 3 ステップにスケジューリングされるだけであるので、3 ステージに分けるためには、ループ処理の 1 ステップが 1 ステージ、2 ステップが 2 ステージ、3 ステップが 3 ステージに割り当てられる。

【0038】

また、例えばパイプライン化されないループ処理が 4 ステップにスケジューリングされ、それを 2 ステージに分ける場合には、1 および 2 ステップが 1 ステージに、3 および 4 ステップが 2 ステージに割り当てられる。この場合、1 ステージを実行するために 2 サイクルが必要となる。

【0039】

パイプライン化された回路では、図 16 に示すような各処理動作が行われる。

【0040】

図 16 に示すように、まず、クロックの 1 サイクル目では $f(1)$ のみが実行される。次に、2 サイクル目では、 $g(1)$ が実行されるのと同時に、2 回目のループ処理の繰り返しが開始されて $f(2)$ が実行される。さらに、3 サイクル目では、 $h(1)$ が実行されて 1 回目のループ処理の繰り返しが終了するのと同時に、2 回目の繰り返しの $g(2)$ と 3 回目の繰り返しの $f(3)$ とが同時に実行される。以降、 k サイクル目では $h(k-2)$ 、 $g(k-1)$ および $f(k)$ が同時に実行される。さらに、11 サイクル目では $h(9)$ と $g(10)$ が、12 サイクル目では $h(10)$ のみが実行されて、ループ処理が終了する。

【0041】

このように、図 14 の動作記述を実行するために、ループ処理がパイプライン化されない場合には 30 サイクル必要であったが、ループ処理がパイプライン化された場合には 12 サイクルでよい。

【0042】

図 14 の動作記述のループ処理を、従来の方法によってパイプライン化した場合の CDFG を図 17 に示している。図 17 の CDFG では、スケジューリングも行われており、各節点が各ステップに分けられている。

【0043】

図 17 に示すように、CDFG において、ステップ 1 では $f(1)$ のみが実行され、ステップ 2 では $g(1)$ と $f(2)$ とが同時に実行される。

【0044】

次に、ステップ 3 に含まれるループ処理部 21 の処理に進む。なお、このループ処理部 21 では、1 回の処理を繰り返すために 1 サイクルを必要とし、ループ処理の繰り返し回数に応じたサイクル数を必要とする。したがって、ステップ 4 の処理以降は、ステップ数とサイクル数とは異なる。

【0045】

ループ処理部 21 内には、1～3 の各ステージの処理が並列に配置されている。節点 22 は、ループ変数 i の初期値 3 を与える節点である。節点 23 および 24 はそれぞれ、入力される値を 1 および 2 減算する節点である。したがって、1 回目のループ処理では、節点 25、26 および 27 によって、 $f(3)$ 、 $g(2)$

）および h（1）が同時に実行される。また、2 回目のループ処理では、f（4）、g（3）および h（2）が同時に実行され、3 回目から 7 回目のループ処理が実行された後、8 回目のループ処理では、f（10）、g（9）および h（8）が同時に実行される。節点 28 で条件（10 以下）が比較され、その条件に一致した場合にループ処理部 21 の処理が終了され、次のステップ 4 の処理へと進む。

【0046】

さらに、ステップ 4 では g（10）および h（9）が実行され、更に次のステップ 5 では h（10）のみが実行される。

【0047】

図 17 の C D F G 中、ループ処理部 21 の本体に対して、その前に実行される部分 20 の処理はプロローグ部と言われ、その後に実行される部分 29 の処理部はエピローグ部と言われている。

【0048】

このような C D F G を作成することによって、ループ処理をパイプライン化して、より高速に処理することが可能となる。

【0049】

【非特許文献 1】

" A C - b a s e d S y n t h e s i s S y s t e m , B a c h ,
a n d i t s A p p l i c a t i o n " P r o c e e d i n g s o f t
h e A S P - D A C 2 0 0 1 , 2 0 0 1 (I E E E C a t a l o g N u m
b e r : 0 1 E X 4 5 5 , I S B N : 0 - 7 8 0 3 - 6 6 3 3 - 6)

【非特許文献 2】

" H i g h L e v e l S y n t h e s i s , " K l u w e r A c a
d e m i c P u b l i s h e r s , 1 9 9 2 (I S B N : 0 - 7 9 2 3 - 9 1
9 4 - 2)

【非特許文献 3】

" P e r c o l a t i o n B a s e d S y n t h e s i s " P r o
c e e d i n g s o f D e s i g n A u t o m a t i o n C o n f e r e n

c e 1990, p p. 444-448 (IEEE)

【特許文献1】

特開 2001-229217号公報

【特許文献2】

特開 2001-142937号公報 (図3 (e))

【0050】

【発明が解決しようとする課題】

しかしながら、図17に示すように、従来の方法でループ処理をパイプライン化した場合には、CDFG中のループ処理部の前後にそれぞれプロログ部とエピログ部とがそれぞれ必要とされ、CDFGが複雑化する。このため、このCDFGからハードウェア (回路図) を動作合成すると、生成されるハードウェアの面積が大きくなり、チップの製造コストが高くなるという問題がある。

【0051】

また、このような従来の方法でループ処理をパイプライン化したCDFGでは、プロログ部とエピログ部とは、ループ繰り返しの条件判断が行われずに必ず実行されてしまう。このため、プロログ部の実行が終了する前にループ自体が終了してしまう場合には、正しい動作が行われなくなる。例えば、図14の動作記述の場合、ループ処理の繰り返し回数を1回とするためには、f (1)、g (1) および h (1) のみを実行する必要があるが、2サイクル目で f (2) が実行されてしまう。例えば、f (2) の処理がメモリへの書き込みや外部との通信を行うという処理を含む場合には、メモリに誤った値が書き込まれたり、外部に不要な値が送出されたりするため、回路が誤動作するという問題がある。

【0052】

このような誤動作を防ぐために、ループ処理の繰り返し回数が少ない場合を別に場合分けしたCDFGを導入することも考えられるが、この場合には、さらにCDFGが複雑化し、ハードウェア (回路図) 面積が大きくなるという問題がある。

【0053】

本発明は、上記従来の問題を解決するもので、従来のプロログ部やエピロー

グ部を必要とせず、ハードウェアの面積増大とコスト増加を防ぐことができる動作合成システム、これを用いた動作合成方法、その各処理手順をコンピュータに実行させる制御プログラム、これが記録されたコンピュータ読み取り可能な可読記録媒体、それらを用いた論理回路の製造方法および論理回路を提供することを目的とする。

【0054】

【課題を解決するための手段】

本発明の動作合成システムは、ループ処理および非ループ処理を含む動作記述から、各種処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフロログラフを生成するコントロールデータフロログラフ生成手段と、このコントロールデータフロログラフを用いてレジスタトランスファレベルのハードウェア構成を自動合成する手段とを有した動作合成システムにおいて、コントロールデータフロログラフ生成手段は、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の1回毎に各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフロログラフのループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれステージ毎の各節点にそれぞれ出力可能とするループ制御部が設けられたものであり、そのことにより上記目的が達成される。

【0055】

また、好ましくは、本発明の動作合成システムにおけるループ制御部は、各制御信号をそれぞれステージ毎の各節点にそれぞれ出力する複数のポートと、複数のポートのうち所定の一または複数のポートに接続されて終了判定を行う終了条件比較節点とを有する。

【0056】

さらに、好ましくは、本発明の動作合成システムにおけるループ処理部は、所定の入力初期値が入力され、これを1サイクル毎にインクリメントした値を順次出力するループ用ポートと、ループ用ポートからの出力値に対してループ処理の回数が規定回数に達したかどうかを判定する判定出力をループ制御部に出力する

ループ回数判定節点と、ステージ毎の各節点とを有する。

【0057】

さらに、好ましくは、本発明の動作合成システムにおけるループ制御部は、ループ回数判定節点からの判定出力を受けて、ループ制御部の各ポートからステージ毎の各節点にそれぞれ、非ループ処理を行わせる各制御信号をそれぞれ出力する。

【0058】

さらに、好ましくは、本発明の動作合成システムにおいて、ループ制御部の複数のポートは、パイプライン化ステージのステージ段数と同じビット数のシフトレジスタ機能を有している。

【0059】

さらに、好ましくは、本発明の動作合成システムにおける複数のポートは、ステージ毎の各節点にそれぞれ、シフトレジスタ機能によって所定の入力初期値が1サイクル毎に順次シフトされた各制御信号をそれぞれ供給する。

【0060】

さらに、好ましくは、本発明の動作合成システムにおいて、ステージ毎の各節点であって、ループ処理部の外部に対して作用する節点に対して、この節点をループ制御部からの制御信号により制御する。

【0061】

さらに、好ましくは、本発明の動作合成システムにおけるコントロールデータフロログラフ生成手段は、パイプライン化されていないループ処理のコントロールデータフロログラフに基づいて、パイプライン化されたループ処理のコントロールデータフロログラフを生成する。

【0062】

さらに、好ましくは、本発明の動作合成システムにおけるコントロールデータフロログラフ生成手段は、パイプライン化されていないループ処理のコントロールデータフロログラフに対して、ステージ境界部分とグラフの枝が交差する箇所に新たにポートを追加するポート追加手段と、並列処理を示すために各ステージを横方向に配置する横方向配置手段と、横方向に配置されたループ処理部の各ス

ページに対して、繰り返し間データ転送枝を接続処理する枝接続手段と、ループ処理部内にループ制御部を設けるループ制御部追加手段とを有する。

【0063】

さらに、好ましくは、本発明の動作合成システムにおける枝接続手段は、ループ処理部外から与えられる初期値およびループ処理部内で演算された値の何れか一方を選択してループ処理部内の変数に代入するセクタ節点を生成し、ループ制御部追加手段は、その何れの値が代入されるかがループ制御部の制御信号によって制御されるように、ループ制御部とセクタ節点とを接続処理する。

【0064】

本発明の動作合成方法は、ループ処理および非ループ処理を含む動作記述から、演算処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフローグラフを生成するコントロールデータフローグラフ生成工程と、このコントロールデータフローグラフを用いてレジスタトランスファレベルのハードウェア構成を自動合成する工程とを有する動作合成方法において、コントロールデータフローグラフ生成工程は、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分ける工程と、複数回のループ処理の1回毎に各パイプライン化ステージの処理を並行して実行するループ処理部を生成する工程と、ループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を該各節点に出力可能とするループ制御部を生成する工程とを有するものであり、そのことにより上記目的が達成される。

【0065】

本発明の制御プログラムは、請求項11に記載の動作合成方法の各処理手順をコンピュータに実行させるものであり、そのことにより上記目的が達成される。

【0066】

本発明の可読記録媒体は、請求項12に記載の制御プログラムが記録されたコンピュータ読み取り可能なものであり、そのことにより上記目的が達成される。

【0067】

本発明の論理回路の製造方法は、請求項1に記載の動作合成システムにより自動合成された回路構成に基づいて論理回路を製造するものであり、そのことによ

り上記目的が達成される。

【0068】

本発明の論理回路は、複数の論理演算手段による並列演算を繰り返し行うループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれ複数の論理演算手段の少なくとも何れかに出力可能とするループ制御部が設けられており、そのことにより上記目的が達成される。

【0069】

上記構成により、以下に、本発明の作用について説明する。

【0070】

本発明にあつては、ループ処理および非ループ処理が含まれている動作記述から、パイプライン化されたC D F Gを生成する際に、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の1回毎に各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフロログラフのループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を各節点に出力するループ制御部が設けられている。

【0071】

一方、従来の構成では、ループ処理をパイプライン化する場合に、その前後のプロローグ部とエピローグ部とが条件判断無しに必ず実行されるため、プロローグ部の実行が終了する前にループ処理が終了すると、正しい動作が行われぬおそれがある。また、プロローグ部とエピローグ部とがループ処理部とは別に設けられているため、ハードウェアの面積が大きくなる。

【0072】

本発明では、プロローグ部とエピローグ部との機能をループ制御部としてループ処理部内に含めたため、従来のようにプロローグ部とエピローグ部とをループ処理部とは別に設ける必要がなく、誤動作やチップ面積増大を防ぐことができる。

【0073】

また、単純な動作記述の場合には、パイプライン化されたC D F Gを作成することは容易であるが、動作記述が複雑である場合には、パイプライン化されたC D F Gを作成することは困難である。このような場合には、パイプライン化されていないループ処理のC D F Gを変形してパイプライン化されたループ処理のC D F Gを作成することが有効である。

【0074】

ループ処理部内に含まれる節点のうち、ループ処理の外部に対して作用しない節点は、実行されても実行されなくても問題が生じないが、外部に対して作用する節点では、誤動作が生じるおそれがある。したがって、そのような外部に対して作用する節点では、ループ制御部からの出力制御信号によって制御する。

【0075】

また、ループ処理部の外から与えられる初期値およびループ処理部内で演算された値のいずれか一方を選択してループ処理部内の変数に代入するセクタ節点においても、いずれの値を代入するかをループ制御部の出力制御信号によって制御する。

【0076】

さらに、ループ処理および非ループ処理の終了判定についても、ループ制御部からの出力制御信号によって制御することができる。

【0077】

本発明によって設計された論理回路図に基づいて製造された論理回路は、ループ制御部が設けられているので、ループ処理部で誤動作が生じることなく、また、プロローグ部やエピローグ部が不要であるためチップ面積を縮小化することができる。

【0078】

【発明の実施の形態】

以下、本発明の動作合成システムの実施形態について、図面を参照しながら説明する。

【0079】

図1は、本発明の動作合成システムの実施形態における要部構成を示すブロッ

ク図である。

【 0 0 8 0 】

図 1 において、動作合成システム 1 0 0 は、コンピュータシステムにて構成され、各部を制御して制御プログラムを実行する制御部 1 1 1 と、制御プログラムおよびそのデータなどが記録された R O M 1 1 2 と、制御部 1 1 1 の制御時にワークメモリとして働く R A M 1 1 3 と、回路設計に必要な各種データが記録されたデータベース 1 1 4 と、ユーザによる操作指令入力を可能とする操作部 1 1 5 と、各種画像表示が為される表示部 1 1 6 とを有し、ループ処理および非ループ処理を含むアルゴリズムが記録された動作記述から、演算処理を示す節点とデータの流れを示す入出力枝により生成されたコントロールデータフローグラフを用いて、R T L（レジスタトランスファレベル）の所望のハードウェア（回路図）を自動合成（自動設計）する。この場合、R O M 1 1 2 には、F D や C D（光ディスク）などの可読記録媒体から制御プログラムおよびそのデータがインストールされている。

【 0 0 8 1 】

制御部 1 1 1 は C P U（中央演算処理装置）で構成されており、制御プログラムに基づいて、C D F G 生成手段 1 1 1 A と、スケジューリング処理手段 1 1 1 B と、アロケーション処理手段 1 1 1 C と、データパス生成手段 1 1 1 D と、コントローラ生成手段 1 1 1 E とによる各機能を順次実行するようになっている。

【 0 0 8 2 】

C D F G 生成手段 1 1 1 A は、アルゴリズム記述（動作記述）におけるデータの流れを解析し、演算などの各種処理機能を表す節点と、データの流れを表す枝とによって構成される C D F G と称されるモデルを作成する。入力枝は演算などの各種処理機能に与えられるデータ、出力枝は演算結果のデータを表す。また、各節点はそれぞれ、演算の種類に関する情報も有している。また、C D F G 生成手段 1 1 1 A は、ループ処理に含まれる各節点を各パイプライン化ステージに分け 1 回毎のループ処理に対して各パイプライン化ステージの各処理を並行して実行するようにした C D F G のループ処理部内に、ループ処理が行われないうきに非ループ処理を行わせる制御信号を各節点に出力可能とするループ制御部が設け

られる構成である。即ち、ループ処理部内にループ制御部が設けられており、C D F G はループ処理部とループ制御部とを有する。

【0083】

スケジューリング処理手段 111B は、C D F G 中の各節点をいつ実行させるかを決定する処理である。スケジューリング処理手段 111B は、C D F G 中の各節点をいくつかのステップに分ける。一つのステップに含まれる節点は、クロック変化の 1 サイクル中に実行される。

【0084】

アロケーション処理手段 111C は、C D F G 中の枝で表されるデータを格納するレジスタを決定するレジスタ決定手段と、C D F G 中の節点で表される演算機能を、どの演算器を用いて行うかを決定する演算器決定手段とを有している。なお、動作合成システムによっては、このアロケーション処理手段 111C による処理がスケジューリング処理手段 111B による処理前に行われる場合もある。

【0085】

データパス生成手段 111D はデータパスを生成する。

【0086】

コントローラ生成手段 111E はコントローラを生成する。これによって、RTL (レジスタトランスファレベル) のハードウェア (回路図) が自動合成 (自動設計) される。

【0087】

以下に、本発明の C D F G 生成手段 111A におけるループ処理のパイプライン化方法の各実施形態 1, 2 について図面を参照しながら詳細に説明する。

【0088】

(実施形態 1)

本実施形態 1 では、図 14 の動作記述に対して、本発明のループ処理のパイプライン化方法を適用した例について説明する。

【0089】

図 2 は、図 14 の動作記述に対して、本実施形態 1 のループのパイプライン化

方法によりループをパイプライン化したC D F Gを示す図である。

【 0 0 9 0 】

図 2 の C D F G では、図 1 7 に示す従来の方法による C D F G から、プロローグ部 2 0 とエピローグ部 2 9 とからなる非ループ処理が削除され、ループ処理を構成する各パイプライン化ステージに対して、その節点（演算処理手段）のループ処理と非ループ処理と制御する制御信号 V 1 ～ V 3 を出力するループ制御部 3 1 がループ処理部 3 0 に設けられている。このループ制御部 3 1 からの出力信号線（出力枝）はそれぞれ、ループ処理部 3 0 を構成する各ステージ内で演算処理を行う節点に接続されている。また、繰り返し変数「i」の初期値を与える節点 4 1 の値は「1」に設定されている。

【 0 0 9 1 】

ここで、まず、ループ処理部 3 0 におけるループ制御部 3 1 について詳細に説明する。

【 0 0 9 2 】

図 2 において、ループ制御部 3 1 には、パイプライン段数分の個数のループ制御変数が設けられている。この図 2 の例では、パイプライン段数が 3 段であるため、三つのループ制御変数 V 1、V 2 および V 3 が設けられており、それぞれ、ポート 3 5、3 6 および 3 7 に対応している。これらのループ制御変数 V 1、V 2 および V 3 は全て、「0」または「1」を表す 1 ビットの変数である。

【 0 0 9 3 】

制御変数 V 1 はループ処理をパイプライン化したときにステージ 1 に含まれる処理を制御するための制御信号として作用するものであり、制御変数 V 2 はステージ 2 に含まれる処理を制御する制御信号として作用するものであって、制御変数 V 3 はステージ 3 に含まれる処理を制御する制御信号として作用するものである。

【 0 0 9 4 】

制御変数 V 1 には初期値「1」が節点 3 2 から与えられ、それ以外のループ制御変数 V 2 および V 3 には初期値「0」が節点 3 3 および 3 4 からそれぞれ与えられる。また、ループ制御部 3 1 では、制御変数 V 1 および V 2 の値が、次の繰

り返しではV2およびV3に転送されるように、ポート35はポート39へ、ポート36はポート40に接続されている。このループ制御部31のC D F Gは、1ビットのレジスタを三つ直列に接続した構成と同様の構成になっており、ハードウェア（回路図）としては3ビットのシフトレジスタによって実現される。また、ポート38には、比較節点43から終了条件の比較結果出力が入力されるようになっている。

【0095】

また、クロックの各サイクルにおいて、V1、V2およびV3の値は、図2に示すように、V1が「1」のときにのみ節点46でfの演算処理が実行されるように、V2が「1」のときにのみ節点47でgの演算処理が実行されるように、また、V3が「1」のときにのみ節点48でhの演算処理が実行されるように、ポート35、36および37と各節点46、47および48がそれぞれ接続され、各節点46、47および48が制御変数V1、V2およびV3によってそれぞれ駆動制御されるようになっている。これにより、演算f、gおよびhが、図5に示す回数と同じ回数だけ、行われるように制御することが可能となる。

【0096】

次に、節点49について説明を行う。

【0097】

節点49は、ポート36および37に接続され、制御変数V2およびV3が入力されて、V2が「0」、かつ、V3が「1」の場合に、ループ処理部30の処理を終了するように図示しないコントローラに処理終了指示が与えられるようになっている。したがって、図3に示すサイクル12で全処理が終了する。

【0098】

節点43において、ループ処理の回数が規定回数に達したか否かが判定されるのはサイクル10であるが、その後も、図17に示すようなエピログ部29に相当する回数分（2サイクル分）だけ、ループ処理部30にて処理が続行されるようになっている。ループ処理としては10回で終了している。

【0099】

この方法によれば、ループの処理回数が固定値とされていない場合についても

、全く同様に、エピログ部（非ループ処理）も含めた正しい実行回数だけ、ループ処理および非ループ処理を含む処理を実行することができる。

【0100】

図2に示すCDFGにおいて、ループ変数*i*の初期値を与える節点41の値は1に設定されている。これは、図17に示すプロログ部20で実行される*i* = 1、2のときの処理も、ループ処理部30の本体内で行うためである。

【0101】

図14の動作記述において、ループ処理は10回繰り返されるが、C言語のwhileループのように、ループを再度繰り返すか否かをループ内の条件式で判断する場合には、ループが1度だけ実行されて終了する場合も考えられる。例えば、図1に示すCDFGの節点43が別の比較式で表され、ループの1回目の実行で「偽」になるような場合には、1サイクル目で、ポート38に「0」が入力される。

【0102】

この場合、V1、V2およびV3は図4のように変化し、3サイクル目で節点49において、ループの実行が終了される。図4では、V1、V2およびV3は、それぞれ、1サイクルの間だけ1となり、関数*f*、*g*および*h*の演算は、それぞれ1度ずつ実行されることになる。

【0103】

このようにして、本実施形態1のループ制御部31を用いることによって、ループの繰り返し回数が小さい場合についても、正しく回路を動作させることができるようになる。

【0104】

次に、ループ処理部30の節点46、47および48について説明を行う。

【0105】

節点46には、二つの値が入力されている。一つはポート42からの*i*の値であり、もう一つはポート35からのV1の値である。節点46は、V1の値が「1」のときにのみ、*f* (*i*) の処理を実行するようになっている。

【0106】

ここで、 f が加算や乗算など、その結果が外部に対して作用しない処理の場合には、 $V1$ の値が「0」のときに f を実行したとしても、その演算結果が使用されないだけであり、実害は生じない。したがって、このような場合には、 $V1$ によって節点 46 の実行を制御する必要はなく、図 17 の節点 25 の場合と同様に、常に f の処理を実行する節点としてもよい。

【0107】

一方、 f がメモリ書き込みや通信など、外部に対して作用する処理の場合には、 $V1$ が「1」のときのみ、その処理が実行されるようにする必要がある。 $V1$ が「0」のときに f の処理が実行されると、外部からみたときにループの振舞いが変わってしまうからである。

【0108】

具体的には、メモリ書き込みの場合にはイネーブル信号に対して $V1$ との論理積を求め、通信の場合には通信要求信号に対して $V1$ との論理積を求めるというような、簡単な回路を追加することによって対応することができる。以上のことは節点 47 および 48 についても、節点 46 の場合と同様である。

【0109】

次に、節点 50 および 51 について説明を行う。

【0110】

この節点 50 および 51 は、 $f(i)$ の処理に並行して、 $g(i-1)$ および $h(i-2)$ の処理を行うために設けられている。例えば、 $f(3)$ と同時に $g(2)$ および $h(1)$ が実行される。 i が「1」のときには、ループ制御変数 $V2$ と $V3$ が「0」であるため、 $f(1)$ は実行されるが、同時に $g(0)$ および $h(-1)$ が実行されることはない。

【0111】

なお、上述したように、 g および h がループ外に対して作用しない処理である場合には、 $g(0)$ および $h(-1)$ が実行されても何ら実害は生じないため、ループ制御変数 $V2$ および $V3$ によって制御する必要もない。

【0112】

また、 i が 12 の場合には、 $V1$ および $V2$ が「0」であるため、 $h(10)$

のみが実行され、 $f(12)$ および $g(11)$ は実行されない。同様に、 f および g が外部に対して作用しない処理であれば、 $f(12)$ および $g(11)$ を実行してもよい。

【0113】

(実施形態2)

本実施形態2では、図5の動作記述に対して、本発明のループ処理のパイプライン化方法を適用した例について説明する。

【0114】

図5はループ処理を含む動作記述の他の例を示す図である。図5において、この動作記述は、 $i = 0$ から i を一つずつインクリメントしながら $i < 10$ の条件を満たす間、配列 a に対して i を書き込み、 i と j を加算した結果を配列 b に対して書き込むという処理を表している。 j の初期値は5であり、次の繰り返しでは $i + j$ が j として用いられる。

【0115】

図6は、図5の動作記述について、ループ処理をパイプライン化していない場合のCDFGを示す図である。

【0116】

図6において、矩形60はループ（ループ処理部60）を表し、ループ処理部60内のCDFGが繰り返し実行される。矩形60の上端に示すポート63および64はそれぞれ変数 i および j の値を表している。ポート63に格納された変数 i の値は、インクリメント演算節点67でインクリメントされて+1加えられた値となり、枝71を介してポート73に至る。ポート73に至ったデータは、次の繰り返しで再びポート63に戻されて変数 i として使用される。

【0117】

節点65では終了条件の比較が行われ、真 ($i \leq 10$) であれば「1」、偽であれば「0」が出力される。また、節点66では、入力が「0」の場合に、図示しないコントローラに対してループ処理部60の処理を終了し、次の状態に移行するように指示が与えられる。

【0118】

節点 6 1 および 6 2 は定数を出力する節点であり、ループ変数 i および j の初期値「0」および「5」が与えられる。これらの値はループ処理部 6 0 の 1 回目の繰り返しで用いられ、2 回目の繰り返し以降は、ポート 7 3 および 7 4 の値が用いられる。

【0 1 1 9】

節点 6 8 は、配列 a に対する書き込みを表す。この節点 6 8 には、二つの入力 が設けられており、左側の入力は配列のインデックス、右側の入力は書き込まれるデータを表す。ここでは、変数 i が配列 a に書き込まれる。また、節点 6 9 は、加算を表し、二つの入力 i および j の値が加算される。その加算結果は、枝 7 2 を介してポート 7 4 に至り、次の繰り返しで再びポート 6 4 に戻されて変数 j として使用される。節点 7 0 は、配列 b に対する書き込みを表し、左側の入力は配列のインデックス、右側の入力は書き込まれるデータを表す。ここでは、節点 6 9 からの加算結果が配列 b に書き込まれる。

【0 1 2 0】

図 6 の C D F G において、節点 6 7 および 6 8 はステップ 1 に、節点 6 9 はステップ 2 に、節点 7 0 はステップ 3 に、それぞれスケジューリングされている。

【0 1 2 1】

このスケジューリングの場合、1 回の繰り返しのために 3 サイクルを要し、ループを 1 0 回繰り返すには 3 0 サイクル必要である。したがって、ループを 1 0 回繰り返すためには、3 0 サイクルが必要となる。

【0 1 2 2】

ところで、図 1 4 のような単純な動作記述の場合には、図 1 に示すようなパイプライン化された C D F G を作成することは比較的容易であるが、図 5 に示す動作記述のように、各ステップ間でデータの受け渡しがあるような場合には、次の繰り返しへのデータ受け渡しも考慮して、パイプライン化されたときのデータの受け渡しを決定する必要があるため、複雑である。

【0 1 2 3】

このような場合には、図 6 に示すパイプライン化していない場合のループ処理の C D F G を元に、パイプライン化されたループの C D F G を作成する方法が有

効である。この方法について、図 7 を用いて説明する。

【0124】

図 7 は、本実施形態 2 の C D F G のパイプライン化処理手順を示すフローチャートである。

【0125】

コンピュータ（計算）上のデータベース 114 に記憶されたパイプライン化されていないループ処理を含む C D F G に対して、図 7 に示すような方法をプログラム化して適用することによって、パイプライン化されたループ処理の C D F G をコンピュータ（計算機）上（例えば C D F G 生成手段 111A）で自動的に生成することが可能となる。

【0126】

以下に、図 7 に示すループ処理のパイプライン化方法の各ステップについて説明する。この例でも、パイプライン段数は「3」としている。

【0127】

まず、ステップ S 11 の前処理では、図 6 に示すパイプライン化されていない C D F G に対して、図 8 A に示すように、ステージの境界部分とグラフの枝が交差する箇所に新たにポート 81、82、83 および 84 を追加する。パイプライン段数は 3 であるため、図 6 のステップ 1～3 は、それぞれ、図 8 A のステージ 1～3 に対応する。

【0128】

ここで、図 6 に示すループ処理の繰り返し用ポート 73 および 74 に接続されている枝 71 および 72 は、ループの次の繰り返しへのデータ転送のために用いられる枝である。

【0129】

次のステップ S 12 では、同一繰り返し内でのデータ転送についてのみ考慮されるため、枝 71 および 72 は、ここでは一旦削除される。この枝 71 および 72 については、ステップ S 13 で考慮されるようになっている。

【0130】

次に、ステップ S 12 のデータフローグラフの変形処理では、各ステージの処

理が並列に動作するように、ステップ S 1 1 で前処理された各ステージ内の C D F G を、図 8 B に示すように、横に並べる。図 8 B の例では、ステージ 1 からステージ 3 が左から右へと並べられている。

【0131】

図 8 B において、ポート 6 3 および 6 4 はそれぞれ、変数 i および j に対応する。パイプライン化前の図 8 A において、枝 8 5 はポート 6 3 と 8 1 とを接続しており、ポート 8 3 のデータは、次のサイクルでポート 8 1 から使用される。

【0132】

これに対して、パイプライン化された図 8 B では、ポート 6 3 の値は、次のループの繰り返しでポート 8 1 に転送される必要がある。そのため、図 8 B においては、枝 8 5 a によって、ポート 6 3 が、ポート 8 1 ではなく、ポート 8 1 a に接続されている。これにより、ポート 6 3 の変数 i の値は、次の繰り返しでポート 8 1 a からポート 8 1 に転送され、節点 6 9 に入力されることになる。

【0133】

その他の枝についても同様に、次のループの繰り返しでデータが転送されるようにポートが接続される。例えば、図 8 A のポート 6 4 と 8 2 とを接続する枝 8 6 は、図 8 B ではポート 6 4 と 8 2 a とを接続する枝 8 6 a とされている。また、図 8 A でポート 8 1 と 8 3 とを接続する枝 8 7 は、図 8 B ではポート 8 1 とポート 8 3 a とを接続する枝 8 7 a とされ、図 8 A で節点 6 9 とポート 8 4 とを接続する枝 8 8 は、図 8 B では節点 6 9 とポート 8 4 a とを接続する枝 8 8 a とされている。

【0134】

次に、ステップ S 1 3 の繰り返し間データ転送枝の接続処理では、図 6 に示すループ処理の繰り返し用ポートに接続されている枝 7 1 および 7 2 に相当する枝 7 1 a および 7 2 a を、図 8 C に示すように接続する。

【0135】

この場合、図 6 に示すパイプライン化されない C D F G において、枝に対してデータを出力する節点と、そのデータを次の繰り返しで使用する節点のステージの差を考慮する。

【0136】

図9は、二つの節点のステージの差に応じて、どのようにパイプライン化すればよいかを示す図である。

【0137】

図9（a）では、ループがパイプライン化されないときに、データが出力されるB節点102とデータが入力されるA節点101とが同一ステージに含まれている場合を示している。このC D F Gがパイプライン化されると、1ステージが1サイクルで実行される場合、各ステージは毎サイクル実行されることになるため、B節点102の処理が実行されて生成されたデータは、1クロックサイクル後に、ループの次の繰り返しのA節点101で使われることになる。これによって、図9（b）に示すパイプライン化されたC D F Gでは、B節点102から出力されたデータが、次のループの繰り返しでA節点に入力されて利用されるように、B節点102をポート104に接続し、次のループの繰り返しでポート104からのデータが供給されるポート103にA節点101を接続する。

【0138】

一方、図9（c）は、ループがパイプライン化されないときに、データが出力されるB節点102がデータが入力されるA節点101よりも1ステージ前にスケジューリングされている場合を示している。このC D F Gがパイプライン化されると、B節点102の処理が実行されて生成されたデータは、2クロックサイクル後に、ループの次の繰り返しのA節点101で使われることになる。これによって、図9（d）に示すパイプライン化されたC D F Gでは、B節点102から出力されたデータが、ループを2回繰り返した後にA節点101に入力されて利用されるように、B節点102を次のステップのポート107に接続し、ポート107からのデータが次のループの繰り返しで入力されるポート105をポート108に接続し、ポート108からのデータが次のループの繰り返しで入力されるポート106にA節点101を接続する。

【0139】

これと同様に、節点Bが節点Aのnステージ前にスケジューリングされている場合には、ループをn+1回繰り返した後にデータが使用されるように、図9（

d) に示すポート 105 および 108 のような繰り返しを追加する。

【0140】

さらに、図 9 (e) は、ループがパイプライン化されないときに、データが出力される B 節点 102 がデータが入力される A 節点 101 よりも 1 ステージ後にスケジューリングされている場合を示している。この C D F G がパイプライン化されると、B 節点 102 と次のループの繰り返しの節点 A 101 とは、同一のクロックサイクルで実行される。これによって、図 9 (f) に示すパイプライン化された C D F G では、B 節点 102 と A 節点 101 とが直接接続される。

【0141】

なお、パイプライン化されないときに、データが出力する B 節点 102 がデータが入力される A 節点 101 よりも 2 ステージ以上後にスケジューリングされている場合には、ループをパイプライン化すること自体が不可能であるため、ここでは考えないこととする。

【0142】

例えば図 6 に示す枝 71 の場合、その枝に対してデータを出力する節点は節点 67 であり、そのデータを次の繰り返して使用する節点も節点 67 であるため、実行されるステージの差は「0」である。したがって、図 9 (b) に示すように枝を接続すればよい。ここでは、図 8 C に示すように、ポート 63、節点 67 および 83a が枝 71a によって接続されている。

【0143】

また、枝 72 の場合にも、その枝に対してデータを出力する節点は節点 69 であり、そのデータを次の繰り返して使用する節点も節点 70 であるため、ステージの差は「0」である。したがって、図 9 (b) に示すように枝を接続すれば良いが、この場合には、接続先のポート 82a に別の枝 86a が接続されている。このため、図 8 C に示すように、セクタ節点 89 を挿入して枝 72a を接続する。このセクタ節点 89 は、1 サイクル目のみ、ポート 64 の値を出力し、それ以外は節点 69 の演算結果をポート 82a に出力するようにコントロールされる。ポート 64 には、図 5 に示す動作記述の変数 j の初期値が入力されており、最初のループの繰り返しでは j の初期値が使用され、それ以降はループ内部で演

算された値である節点 69 からの出力が使用される。

【0144】

次に、図 7 のステップ S14 のループ制御部の追加処理では、図 8D に示すように、C D F G にループ制御部 90 が加えられる。

【0145】

図 8D において、図 6 の C D F G が 3 段にパイプライン化されるため、三つのループ制御変数 V1、V2 および V3 が必要であり、ポート 91、92 および 93 から、V1、V2 および V3 がそれぞれ出力される。

【0146】

制御変数 V1 は、ループがパイプライン化されたときにステージ 1 に含まれる処理を制御する制御信号であり、制御変数 V2 は、ステージ 2 に含まれる処理を制御する制御信号であり、制御変数 V3 は、ステージ 3 に含まれる処理を制御する制御信号である。制御変数 V1 は、初期値として 1 が与えられ、制御変数 V2 および V3 は、初期値として 0 が与えられる。

【0147】

制御変数 V1 および V2 の値は、次の繰り返しでそれぞれ V2 および V3 に与えられるように、ポート 91 とポート 92 a とが枝 94 b を介して接続され、ポート 92 とポート 93 a とが枝 95 b を介して接続されている。

【0148】

また、ポート 91 a には、枝 94 c を介して、比較節点 65 によるループ終了条件の比較結果が入力されており、ループ終了条件が「真」になったときに、ポート 91 a に「0」が入力される。

【0149】

さらに、終了を判定する節点 97 は、枝 95 c および 96 b を介して、ポート 92 およびポート 93 に接続されており、制御変数 V2 が「0」、かつ、V3 が「1」のときにループの処理を終了するように、図示しないコントローラに指示が与えられる。

【0150】

ポート 91 および 93 からの制御変数 V1 および V3 は、それぞれ、枝 94 a

および 96 a を介してステージ 1 の節点 68 およびステージ 3 の節点 70 に接続されている。制御変数 V1 の値は、図 3 に示すように、1 サイクル目～10 サイクル目が「1」であり、11 サイクル目および 12 サイクル目は「0」である。したがって、1 サイクル目～10 サイクル目では、節点 68 において配列 a に対する書き込み処理が行われ、11 サイクル目および 12 サイクル目では、節点 68 の処理は行われない。また、制御変数 V3 の値は、図 3 に示すように、1 サイクル目および 2 サイクル目が「0」であり、3 サイクル目～12 サイクル目は「1」である。したがって、1 サイクル目および 2 サイクル目では節点 70 の処理は行われず、3 サイクル目～12 サイクル目では、節点 70 において配列 b に対する書き込み処理が行われる。このように、配列 a に書き込みが行なわれる節点 68 および配列 b に書き込みが行なわれる節点 70 に対して、プロログ部およびエピログ部の動作を制御することができる。

【0151】

また、ポート 92 からの制御変数 V2 は、枝 95 a を介して、セクタ節点 89 に入力されている。制御変数 V2 の値は、図 3 に示すように、1 サイクル目が「0」で、2 サイクル目から 11 サイクル目は「1」である。したがって、1 サイクル目は、セクタ節点 89 において左の入力に接続されるポート 64 の値が出力され、2 サイクル目以降は右の入力に接続される節点 69 の演算結果が出力される。このように、1 サイクル目のみポート 64 に接続される外部からの初期値が選択され、それ以降は、ループ内部で演算が行われた結果が選択される。制御変数 V2 は、12 サイクル目が「0」であるため、12 サイクル目はポート 64 の値が選択されるが、12 サイクルでループの処理が終了されるため、12 サイクル目のセクタ節点 89 の動作は、回路全体の動作に対して無関係である。

【0152】

即ち、C D F G 生成手段 111 A の他の実施形態として、パイプライン化されていない C D F G に対して、ステージ境界部分とグラフの枝が交差する箇所に新たにポートを追加するポート追加手段と、並列処理を示すために各ステージを横方向に配置する横方向配置手段と、横方向に配置された C D F G に対して、繰り返し間データ転送枝を接続処理する枝接続手段と、ループ処理部にループ制御部

を追加処理するループ制御部追加手段とを有している。

【0153】

以上のような処理手順により、パイプライン化されていないループのCDFGから、ループ処理がパイプライン化されたCDFGを得ることができる。

【0154】

このように、ループ処理がパイプライン化されたCDFGを生成し、図10に示すように、アロケーション処理、データパス生成処理、コントローラ生成処理などを行うことによって、動作記述を実現するためのハードウェアを合成することが可能である。

【0155】

したがって、上記実施形態1, 2によれば、ループ処理をパイプライン化するときに、ループ処理部30または60を構成する各パイプライン化ステージに対して、そのパイプライン化ステージに含まれる各節点の処理が実行されるクロックサイクルと、非ループ処理が実行されるクロックサイクルとを制御する制御信号V1～V3を各節点46～48または68～70に出力するループ制御部31または90を設けたCDFGを作成する。これによって、動作記述からハードウェア（設計回路図）を合成する動作合成において、少ない面積の増加でCDFG中のループ処理をパイプライン化することができる。

【0156】

【発明の効果】

以上により、本発明によれば、ループ処理および非ループ処理が含まれている動作記述から、パイプライン化されたCDFGを生成する際に、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の1回毎に各パイプライン化ステージの処理を並行して実行することを示すコントローラデータフローグラフのループ処理部に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を各節点に出力するループ制御部を設けることによって、従来のプロログ部とエピログ部の非ループ処理を設けなくても、この非ループ処理を、パイプライン化したループ処理部に実行させることができ、従来のループ処理のパイプライン化方法で問題となっているハード

ウェア構成の面積増大やコスト増加を防ぐことができる。また、プロローグ部の処理が終了する前にループ処理が終了されてしまう場合や、ループ処理の繰り返し回数が一定でない場合であっても、誤動作を招くことなく、ループ処理のパイプライン化を行うことができる。

【図面の簡単な説明】

【図 1】

本発明の動作合成システムの実施形態における要部構成を示すブロック図である。

【図 2】

図 1 4 の動作記述に対して、本実施形態 1 のループ処理のパイプライン化方法によりループ処理をパイプライン化した C D F G を示す図である。

【図 3】

図 2 のループ制御部の出力信号の一例を示す図である。

【図 4】

図 2 のループ制御部の出力信号の他の一例を示す図である。

【図 5】

図 1 4 の動作記述とは別の動作記述の一例を示す図である。

【図 6】

パイプライン化されない C D F G の一例を示す図である。

【図 7】

図 6 の C D F G をパイプライン化する本実施形態 2 のパイプライン化処理手順を示すフローチャートである。

【図 8 A】

C D F G のパイプライン化処理手順（その 1）を示す図である。

【図 8 B】

C D F G のパイプライン化処理手順（その 2）を示す図である。

【図 8 C】

C D F G のパイプライン化処理手順（その 3）を示す図である。

【図 8 D】

C D F G のパイプライン化処理手順（その 4）を示す図である。

【図 9】

（a）および（b）～（e）および（f）は、繰り返し間枝の接続処理例を説明するための図である。

【図 10】

従来の動作合成方法の処理手順を示すフローチャートである。

【図 11】

従来の動作記述の式の一例を示す図である。

【図 12】

図 11 の動作記述に対応した C D F G を表す図である。

【図 13】

図 12 の C D F G のデータ構造を説明するための図である。

【図 14】

動作記述の一例を示す図である。

【図 15】

従来のパイプライン化されない C D F G の一例を示す図である。

【図 16】

従来のパイプライン動作について説明するための図である。

【図 17】

従来手法によりパイプライン化された C D F G の一例を示す図である。

【符号の説明】

30、60 ループ処理部

32、33、34、41、61、62 定数出力節点

42、45、63、64、63a、64a、73、74、81、82、81a
、82a、83、84、83a、84a ループ処理部のポート

43、65 終了条件比較節点

44、67 インクリメント演算節点

49、66 終了判定節点

46～48 関数演算節点

20 プロローグ部

50、51 減算節点

29 エピローグ部

31、90 ループ制御部

35~40、91~93、91a~93a ループ制御部のポート

68、70 配列書き込み節点

69 加算節点

71、72、71a、72a、85~88、85a~88a 枝

81~84 ステージの境界部分とC D F Gの枝が交差する箇所のポート

89 セレクタ節点

94a、94b、94c、95a、95b、95c、96a、96b ループ

制御部に接続された枝

100 動作合成システム

101、102 節点

103~108 ポート

111 制御部

111A C D F G生成手段

111B スケジューリング手段

111C アロケーション手段

111D データパス生成手段

111E コントローラ生成手段

112 ROM

113 RAM

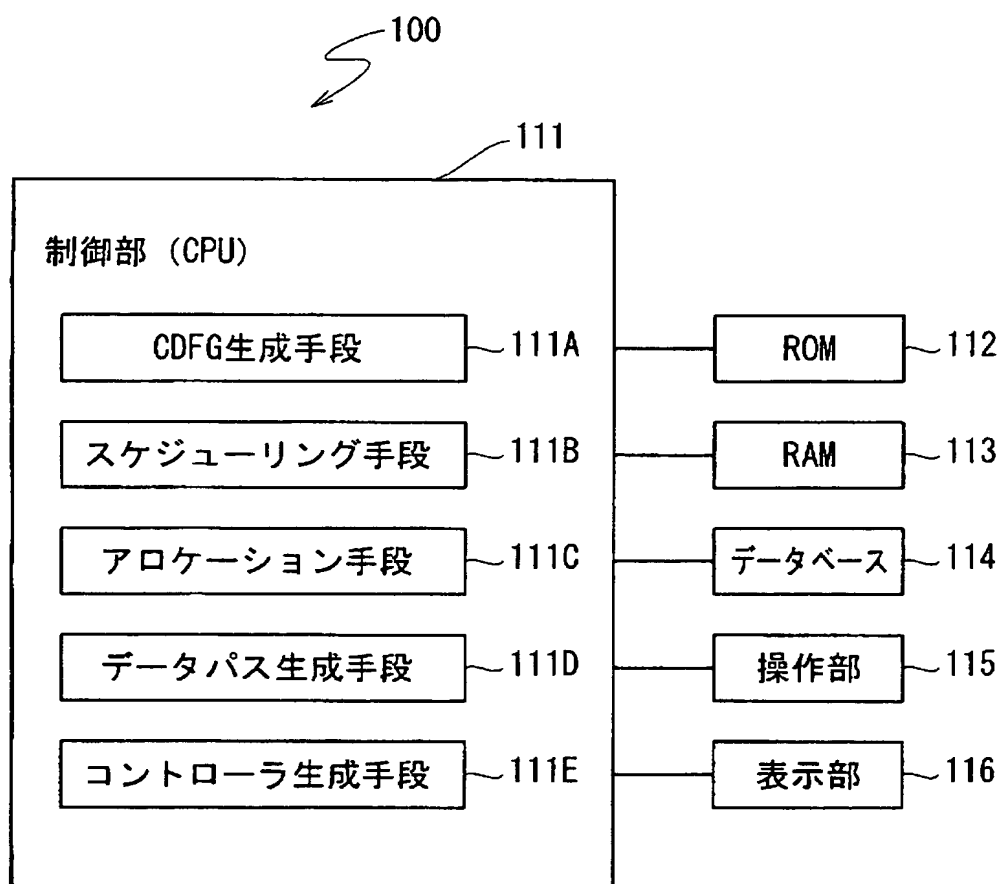
114 データベース

115 操作部

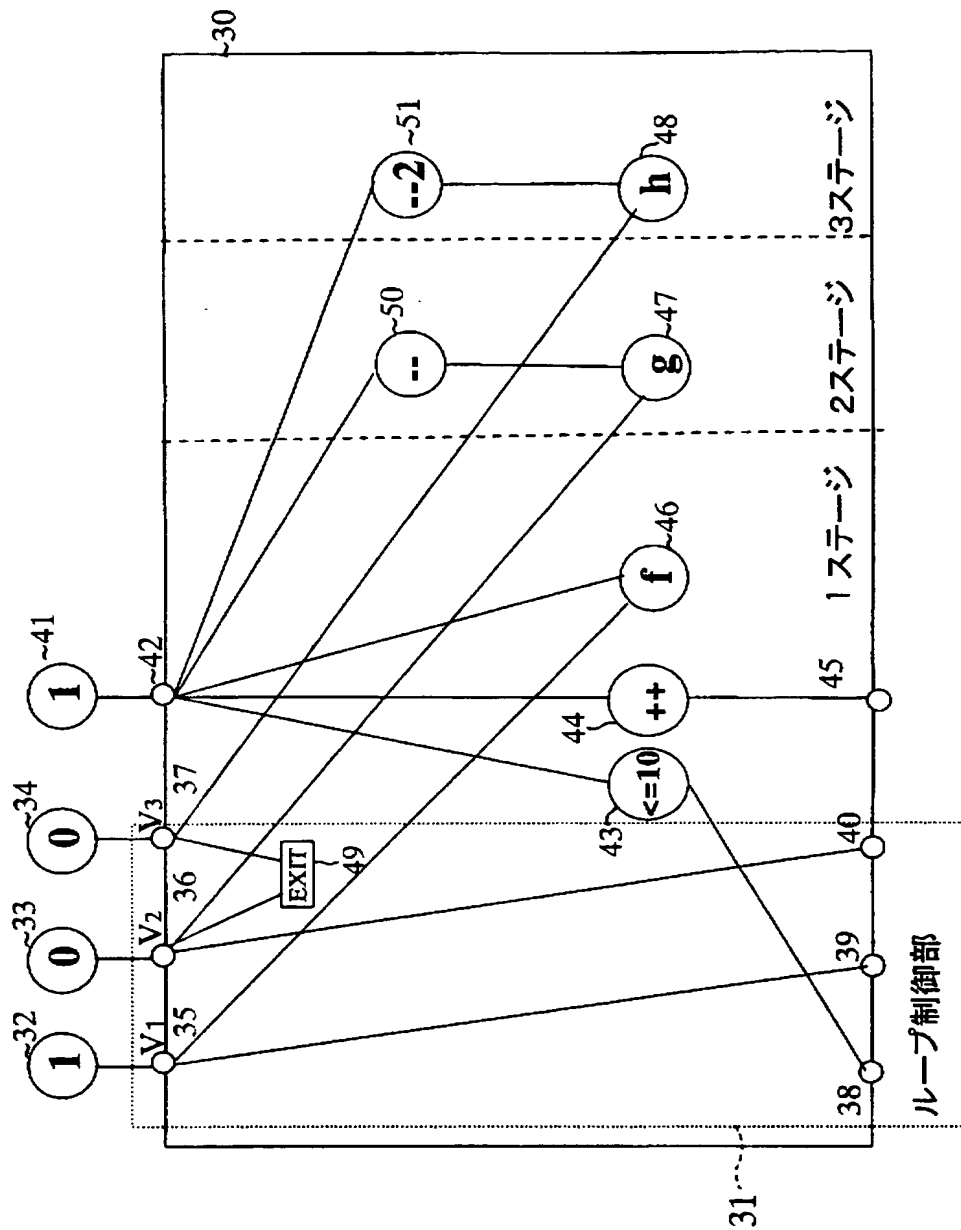
116 表示部

【書類名】 図面

【図 1】



【図 2】



【図 3】

サイクル	V1	V2	V3
1	1	0	0
2	1	1	0
3~10	1	1	1
11	0	1	1
12	0	0	1

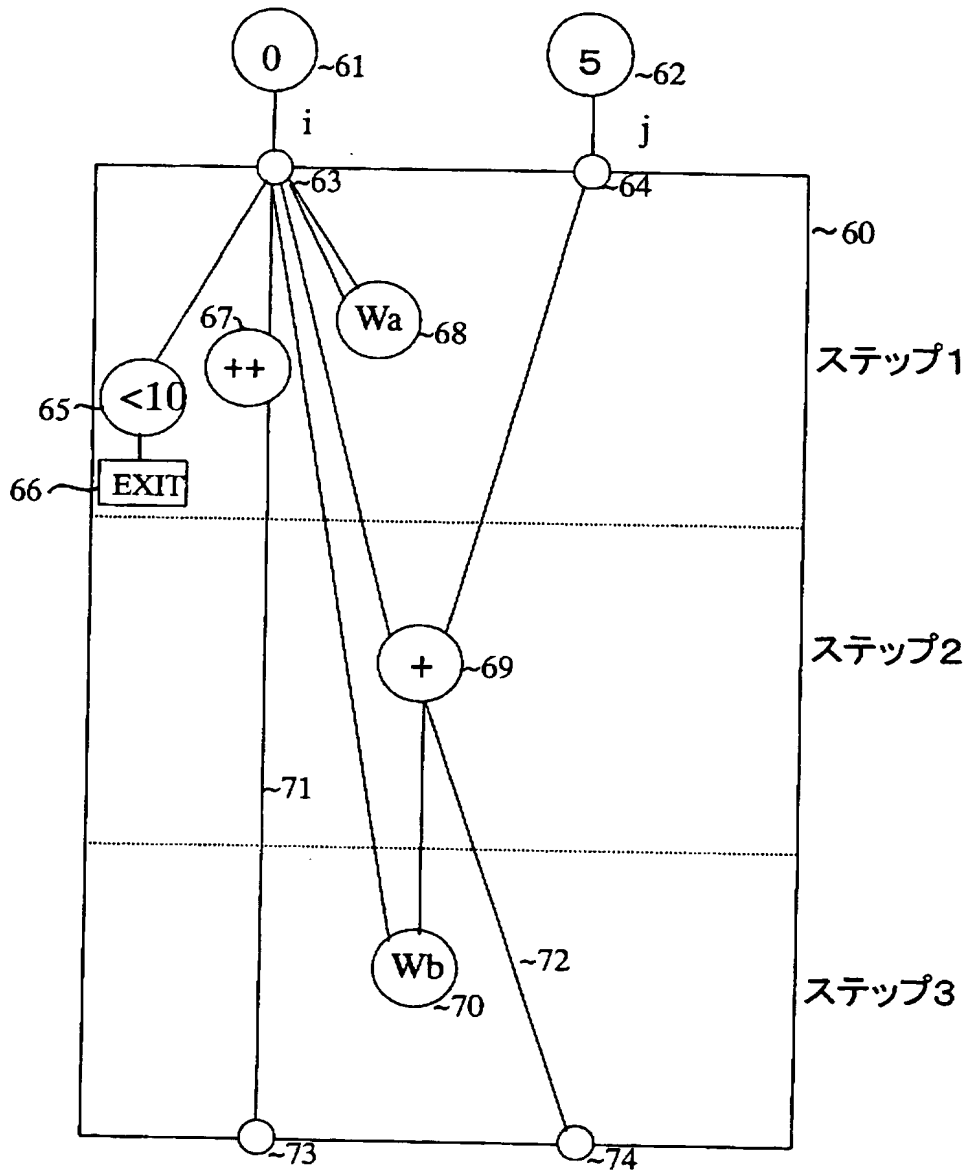
【図 4】

サイクル	V1	V2	V3
1	1	0	0
2	0	1	0
3	0	0	1

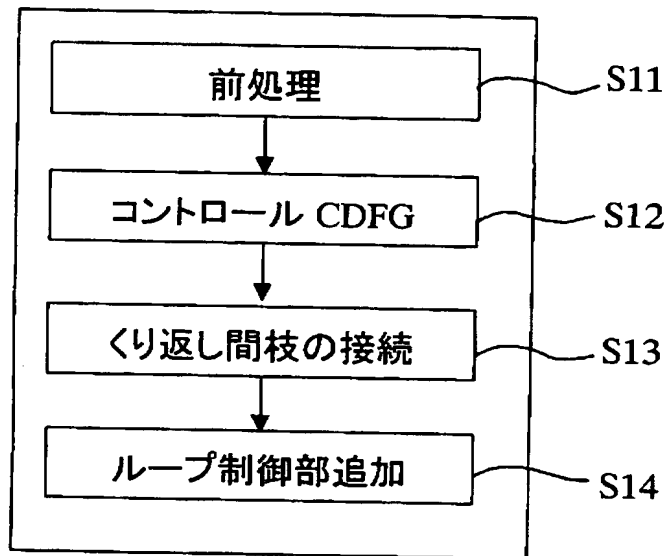
【図 5】

```
j = 5;  
for (i = 0; i < 10; i++)  
{  
    a[i] = i;  
    j += i;  
    b[i] = j;  
}
```

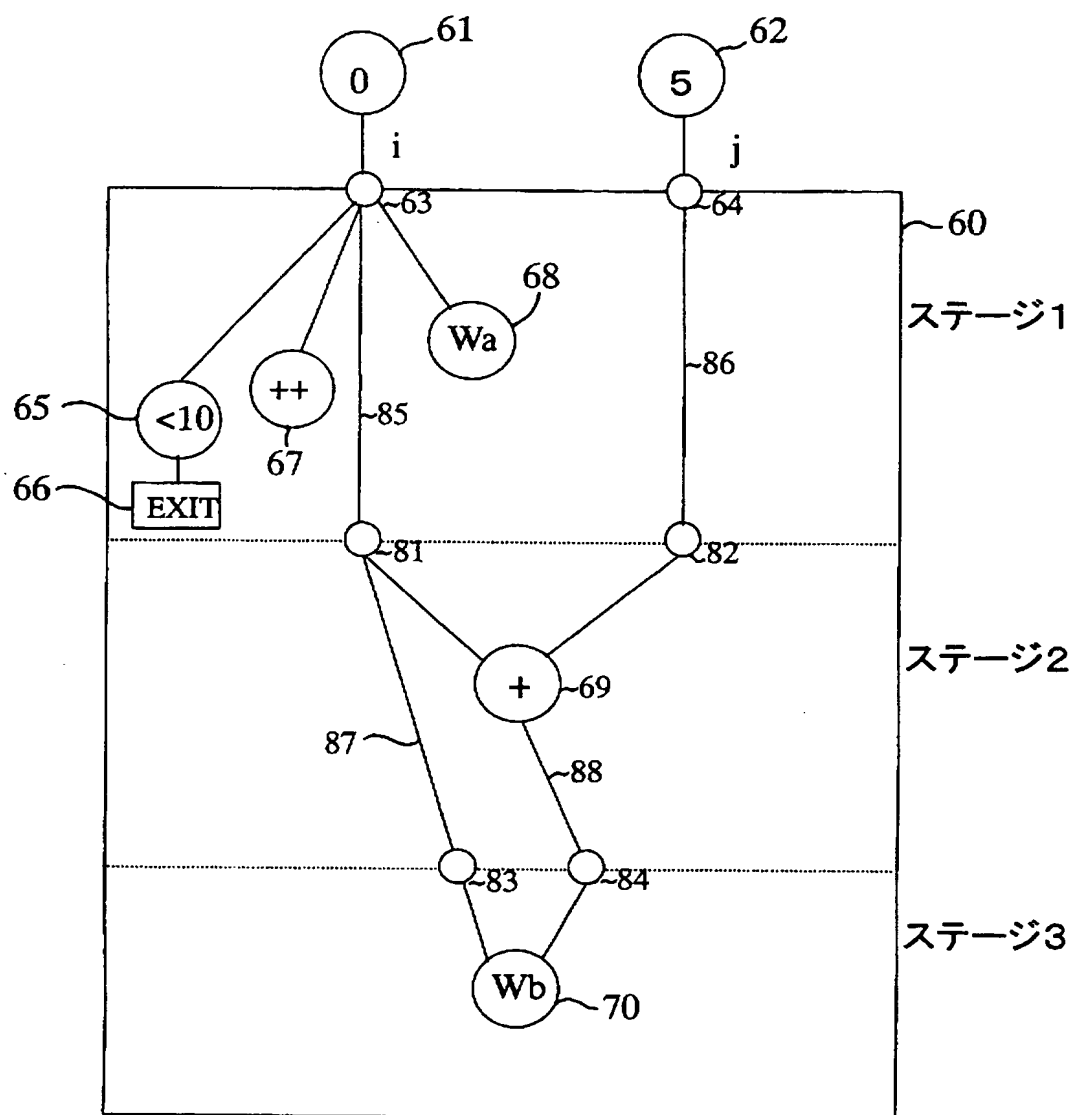
【図 6】



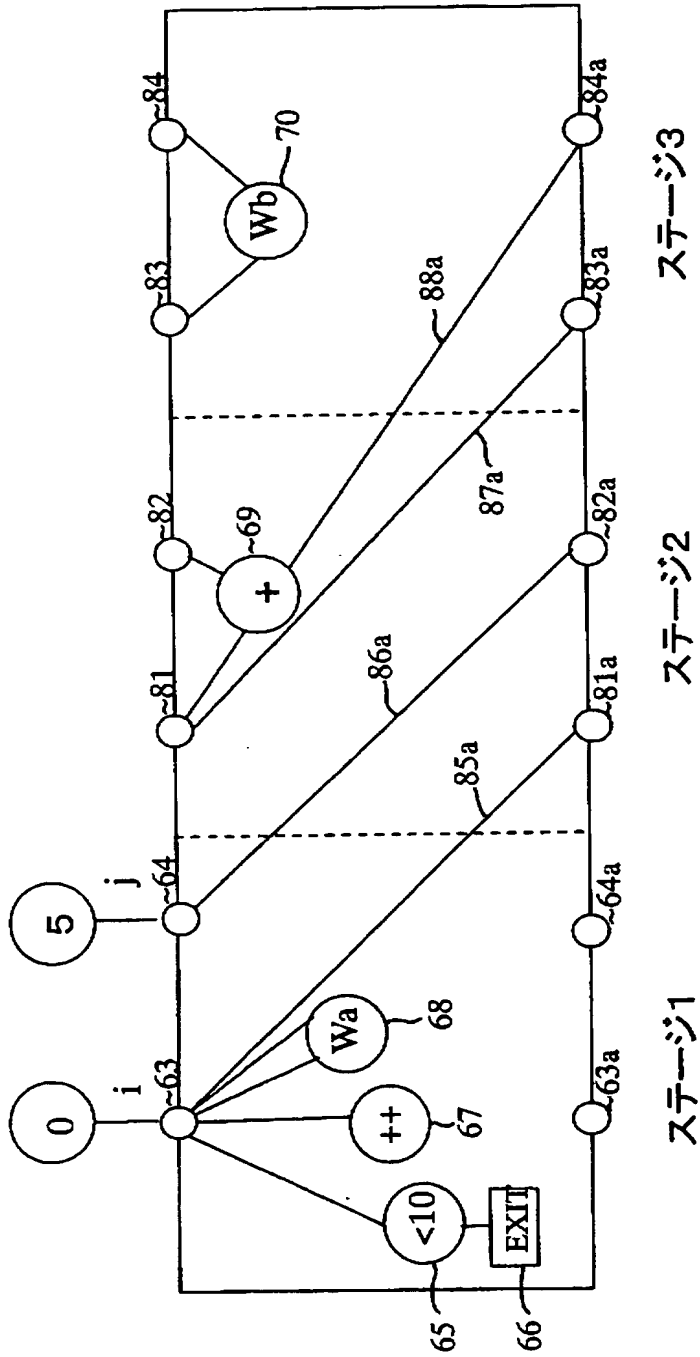
【図 7】



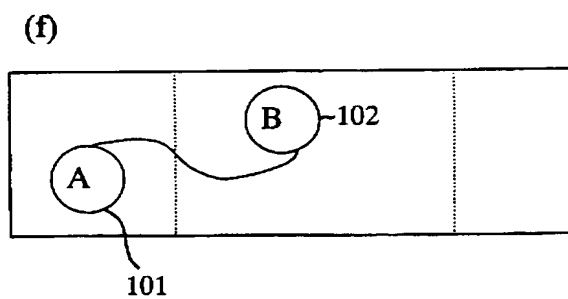
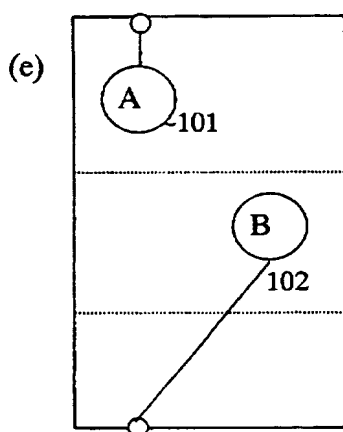
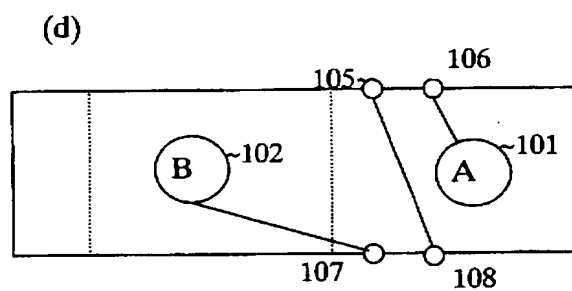
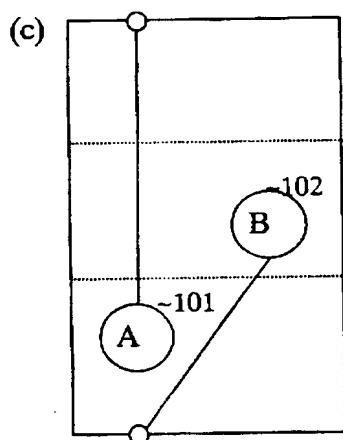
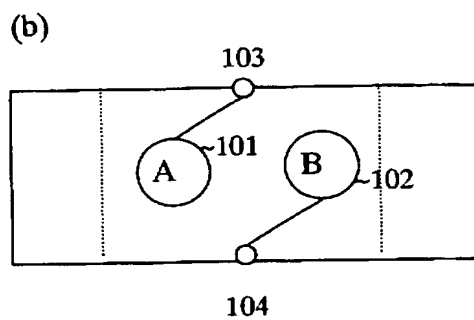
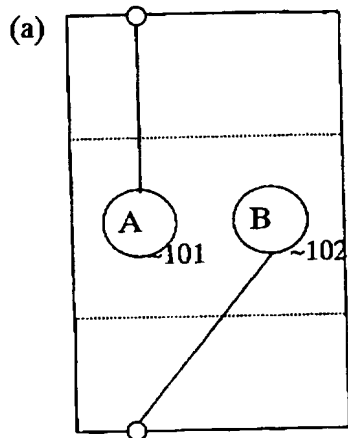
【図 8 A】



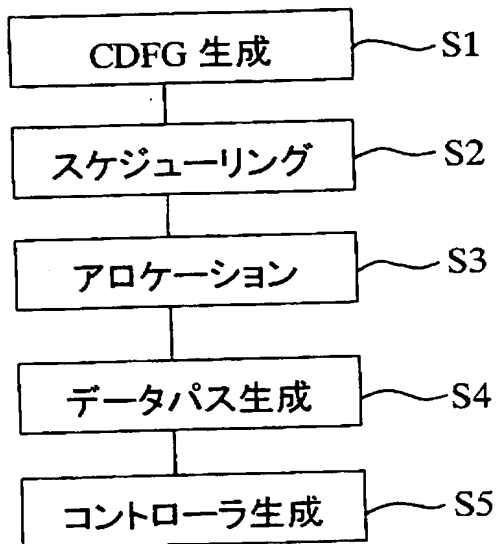
【図 8 B】



【図 9】



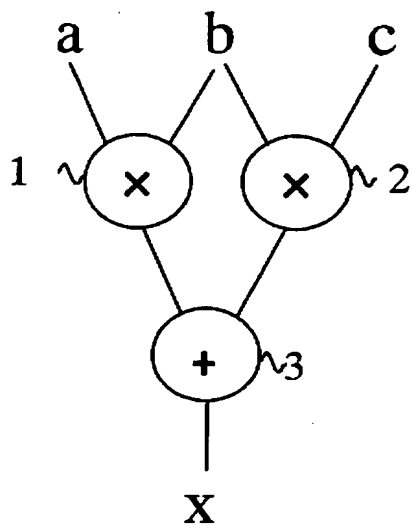
【図 10】



【図 11】

$$x = a \times b + b \times c$$

【図 12】



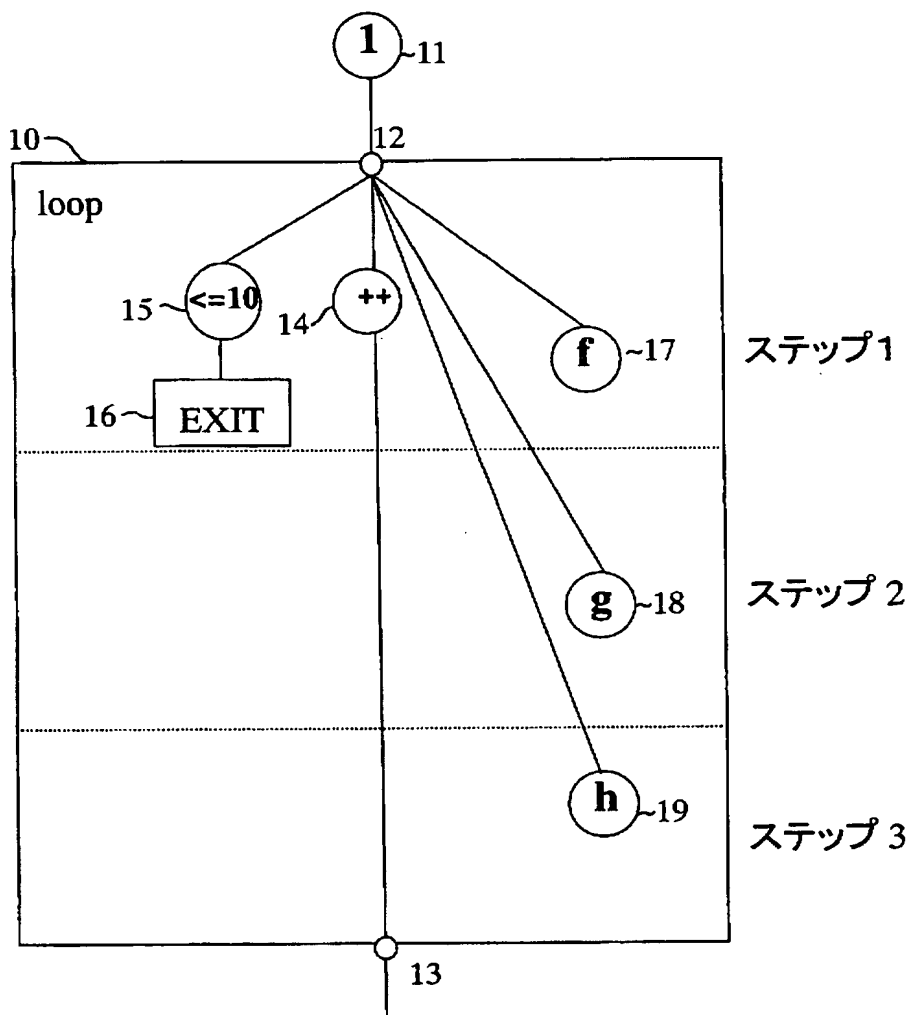
【図 1 3】

```
struct Node
{
    int  node_id;
    int  in_edge[2];
    int  out_edge[1];
    int  op_type;
}
struct Edge
{
    int  edge_id;
    int  from_node;
    int  to_node;
}
```

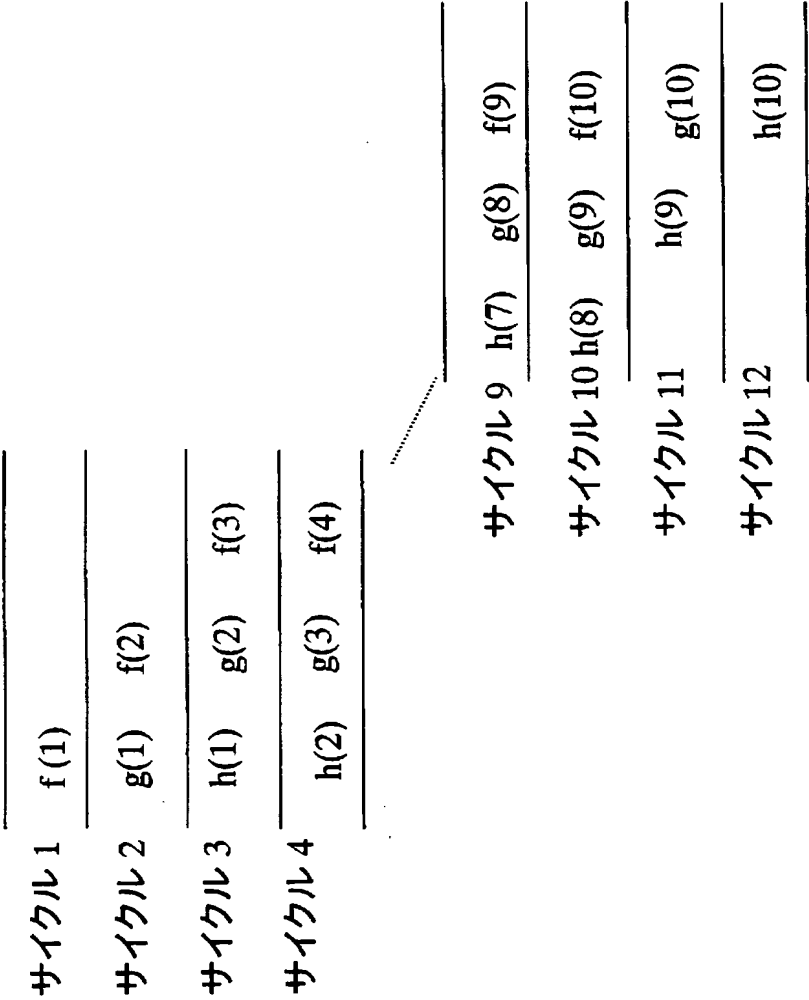
【図 1 4】

```
for( i =0; i <=10 ; i++)
{
    f(i);
    g(i);
    h(i);
}
```

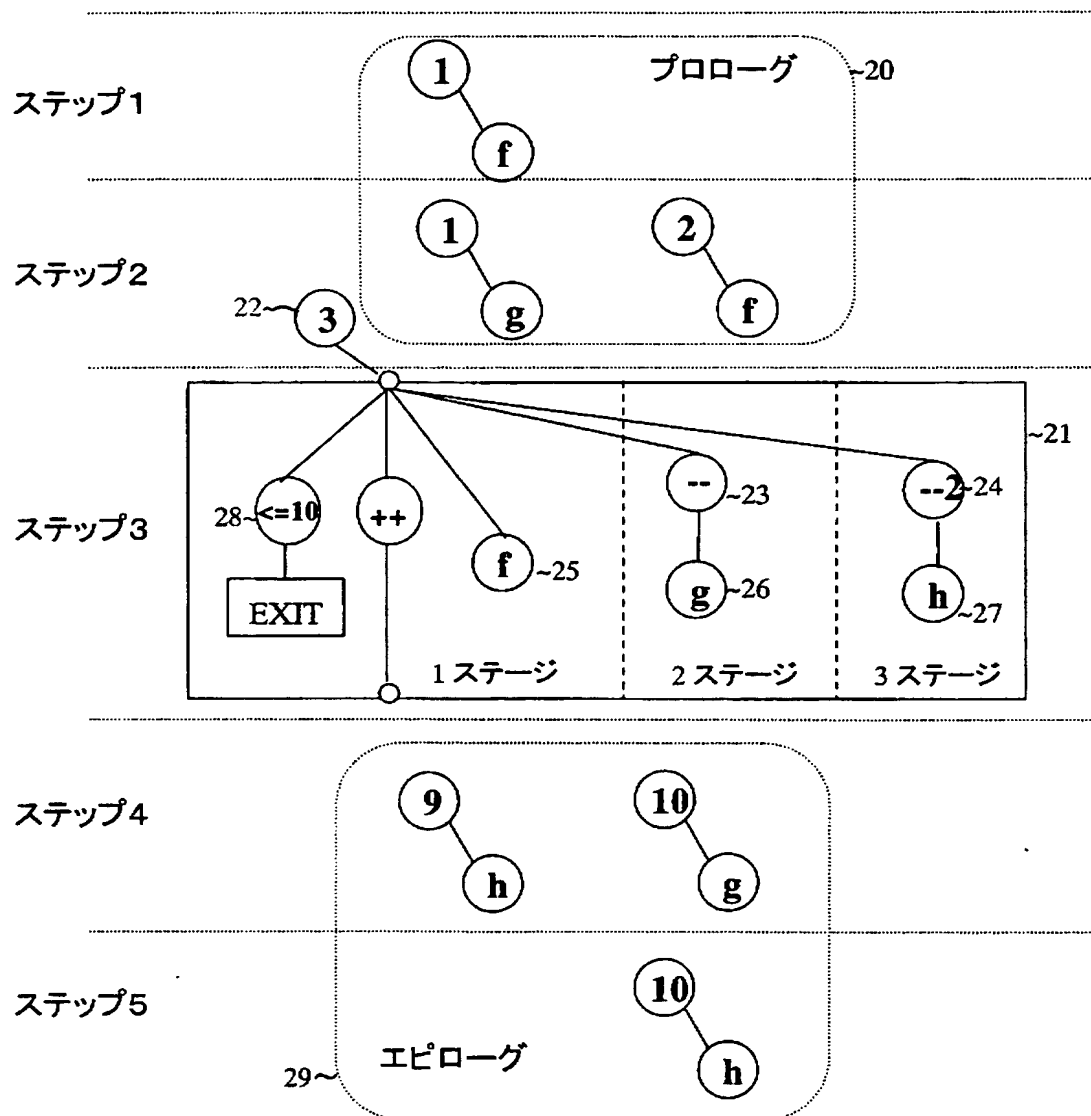
【図 15】



【図 1 6】



【図 17】



【書類名】 要約書

【要約】

【課題】 動作記述からハードウェアを合成する動作合成において、少ない面積の増加で C D F G 中のループ処理をパイプライン化する。

【解決手段】 ループ処理をパイプライン化するときに、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の 1 回毎に各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフローグラフのループ処理部 3 0 に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を各節点に出力するループ制御部 3 1 を設けた C D F G を生成する。

【選択図】 図 1

認定・付加情報

特許出願の番号	特願 2003-120600
受付番号	50300690792
書類名	特許願
担当官	小野寺 光子 1721
作成日	平成 15 年 4 月 25 日

<認定情報・付加情報>

【特許出願人】

【識別番号】 000005049

【住所又は居所】 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号

【氏名又は名称】 シャープ株式会社

【代理人】 申請人

【識別番号】 100078282

【住所又は居所】 大阪市中央区域見 1 丁目 2 番 2 7 号 クリスタル
タワー 1 5 階

【氏名又は名称】 山本 秀策

【選任した代理人】

【識別番号】 100062409

【住所又は居所】 大阪府大阪市中央区域見 1 丁目 2 番 2 7 号 クリ
スタルタワー 1 5 階 山本秀策特許事務所

【氏名又は名称】 安村 高明

【選任した代理人】

【識別番号】 100107489

【住所又は居所】 大阪市中央区域見一丁目 2 番 2 7 号 クリスタル
タワー 1 5 階 山本秀策特許事務所

【氏名又は名称】 大塩 竹志

次頁無

特願 2 0 0 3 - 1 2 0 6 0 0

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 0 4 9]

1 . 変 更 年 月 日

1 9 9 0 年 8 月 2 9 日

[変 更 理 由]

新 規 登 録

住 所

大 阪 府 大 阪 市 阿 倍 野 区 長 池 町 2 2 番 2 2 号

氏 名

シャープ株式会社